15400 Calhoun Drive, Suite 400
Rockville, Maryland, 20855
(301) 294-5200
http://www.i-a-i.com

Intelligent Automation Incorporated

Enhancements for a Dynamic Data Warehousing and Mining System for Large-scale HSCB Data

Progress Report No. 1

Reporting Period: March 22, 2016 – April 21, 2016

Contract No. N00014-16-P-3014

Sponsored by
ONR, Arlington VA
COTR/TPOC: Dr. Rebecca Goolsby

Prepared by
Onur Savas, Ph.D.

Progress Report No. 1

Enhancements for a Dynamic Data Warehousing and Mining System Large-Scale HSCB Data

Submitted in accordance with requirements of
Contract #N00014-16-P-3014

Performance period: March 22, 2016 to April 21, 2016
(PI: Dr. Onur Savas, 301.294.4241, osavas@i-a-i.com)

1 Work Performed within This Reporting Period

In this reporting period, we performed the following tasks.

- **Further enhanced the graph database architecture for modeling, storing and querying Twitter interaction graphs.** We have developed an architecture to ingest Twitter data, convert it into Twitter interaction graphs, and store in a graph database, namely OrientDB.

- **Refined the k-hop neighborhood queries for Twitter interaction graphs.** We have refined the implementation and design of k-hop neighborhood queries, i.e., up to and including k distance neighbors, and matured a web-based visualization and UI to interact with the graph database.

1.1 Storing Twitter Interaction Graphs in OrientDB

The first step in developing the querying capability is to store a graph efficiently. A graph database, for all practical purposes, can be represented as the graph itself. In its simplest form, we consider the Twitter interaction graph modeled as an undirected graph $G = (V, E)$, where V is the set of nodes (vertices) and E is the set of edges. For a collection of tweets $\{\tau(\theta) | \theta \in \mathbb{Z}^+\}$, where each tweet $\tau(.)$ can be uniquely identified by its unique *tweet ID* $\theta \in \mathbb{Z}^+$, let $\tau(\theta)$ be tweeted by user $u_{\tau(\theta)}$ and let $u_{\tau(\theta)}$ have retweeted, mentioned, or replied to K_θ users $\mathcal{I}(\tau(\theta)) = \{v^1_{\tau(\theta)}, v^2_{\tau(\theta)}, \ldots, v^{K_\theta}_{\tau(\theta)}\}$. Of course, if no retweets, mentions, or replies are present, then $\mathcal{I}(\tau(\theta)) = \emptyset$. We can then unambiguously specify the Twitter Interaction Graph G with $V = \{u_{\tau(\theta)} \cup \mathcal{I}(\tau(\theta)) | \theta \in \mathbb{Z}^+\}$ and $E =$

$\{u_{\tau(\theta)} \times \mathcal{I}(\tau(\theta)) | \theta \in \mathbb{Z}^+\} = \quad \{(u_{\tau(\theta)}, v^1_{\tau(\theta)}), (u_{\tau(\theta)}, v^2_{\tau(\theta)}), \ldots, (u_{\tau(\theta)}, v^K_{\tau(\theta)}) | \theta \in \mathbb{Z}^+\}$. One can enhance G by adding other types of interactions, e.g., by adding an edge if a user uses a hashtag in his/her tweet. In this particular case, the vertex set will have hashtags as well. In fact, in this reporting

To implement the graph modeling capabilities above, we used an open source graph database, namely OrientDB (http://www.orientdb.com), and an SQL-like graph querying language. OrientDB provides an NoSQL engine that stores and queries graphs via both (i) a graph database API and document API, and (ii) supports schema-less, schema-full, and schema-mixed modes. Our initial experimentation with OrientDB, an open source graph database, and its SQL-like graph querying language yielded promising results. In particular, we inserted a Twitter Interaction Graph with $|V| > 34k$ and $|E| > 421k$ in less than 6 minutes. In addition, we ordered nodes of this graph by degree in less than 2 seconds. A basic graph service that performs basic graph database management operations, e.g., insert, delete, update, is also implemented.

We have also designed and implemented the *k*-hop neighbor queries. In brief and informally, starting from a node *v*, *k*-hop queries finds the neighbors of *v*, neighbors of neighbors of *v* (a.k.a. second degree neighbors), and so on until *k*-hop neighbors. In particular, for a querying function *Q(.)*, a "2-hop" neighbors query is implemented as follows.

$$Q(v) = \{(v, v') \cup (v', v'') | (v, v') \in E \; and \; (v', v'') \in E \; for \; \forall \, v', v'' \in V\}.$$

This can be generalized to *k*-hop by including neighbors of neighbors of neighbors up *k*-hop.

This querying service has also been implemented using a Groovy/Grails framework with functions written in Java.

1.2 Visualization and UI

To visualize the graph returned by the queries, we have further developed IAI's graph visualization capabilities. We have also incorporated a UI to call the *k*-hop query service. These capabilities are all incorporated to Scraawl.

Figure 1 is a visualization for 4-hop neighborhood query starting from #EasterEggRoll. The Twitter interaction graph has been created from a collection of tweets that have been collected between 28 Mar, 2016 03:00 AM and 29 Mar, 2016 02:59 using the keyword #EasterEggRoll. Overall, 25063 tweets were collected. The graph has 16171 nodes (vertices) and 48594 edges.

The "Neighbors" button to the upper left corner call the k-hop querying service with (i) starting node (in this case #EasterEggRoll), and (ii) the *k*-parameter (in this case *k* = 4).

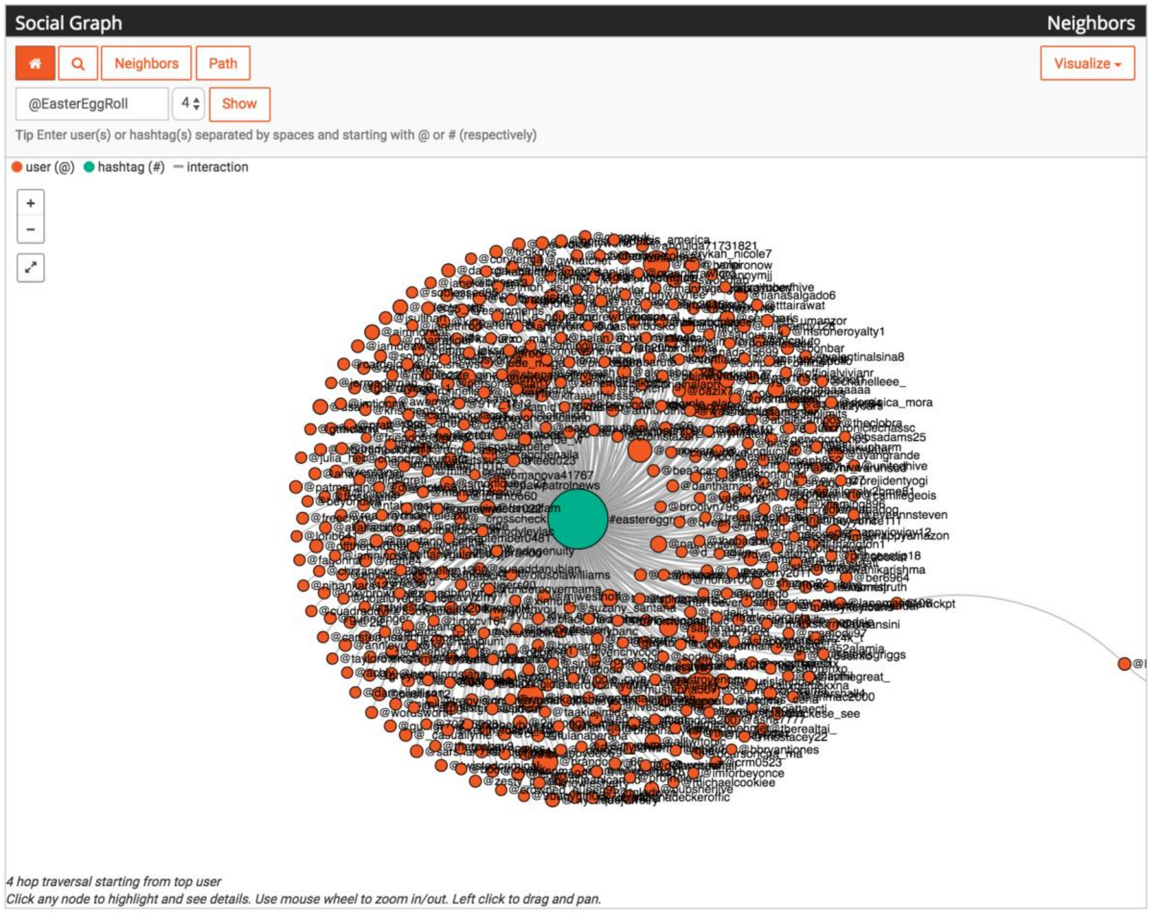
Social Graph
Neighbors
Neighbors
Path
Visualize
@EasterEggRoll
4
Show
Tip Enter user(s) or hashtag(s) separated by spaces and starting with @ or # (respectively)
user (@)
hashtag (#)
interaction
#eastereggroll
4 hop traversal starting from top user
Click any node to highlight and see details. Use mouse wheel to zoom in/out. Left click to drag and pan.

Figure 1: 4-hop neighborhood query starting from "#eastereggroll".

2 Current Problems

None.

3 Work to be Performed in the Next Reporting Period

In the next report period, we will focus on the following tasks:

- We will finish implementation of Task 1.
- We will deliver Scraawl 1.13.0.